

Technologie obiektowe - projekt

Narzędzia odwzorowania obiektowo -
relacyjnego: Hibernate

Grupa:

1ID21A

Skład zespołu:

-Kamil Pazera

-Piotr Nowacki

Spis treści:

- 1) Mechanizm odwzorowania obiektowo – relacyjnego
- 2) Narzędzia odwzorowania obiektowo – relacyjnego
- 3) Metody sterowania
- 4) Sposoby odwzorowania dziedziczenia
- 5) Asocjacje
- 6) Aplikacja
- 7) Wnioski

1. Mechanizm odwzorowania obiektowo – relacyjnego:

Odwzorowanie (mapowanie) obiektowo – relacyjne (ORM) to odwzorowanie obiektowej struktury danych na postać relacyjną, oraz na odwrót. Stosuje się je najczęściej gdy aplikacja jest pisana przy pomocy programowania obiektowego, a baza danych jest relacyjna.

ORM posiada zarówno wady jak i zalety:

Zalety	Wady
<ul style="list-style-type: none">- Operacje na bazie danych są przeprowadzane za pomocą programowania obiektowego, co oferuje mechanizmy niedostępne w relacyjnej bazie danych.- W trakcie tworzenia aplikacji można poświęcić więcej czasu na implementację logiki.- Minimalizuje użycie prostych zapytań SQL.- Łatwość wprowadzania poprawek do logiki.	<ul style="list-style-type: none">- Tworzy dodatkową warstwę, przez którą dane muszą być wysyłane.- Skomplikowane zapytania muszą być dobrze przemyślane i napisane

2. Narzędzia odwzorowania obiektowo – relacyjnego:

Istnieje wiele narzędzi odwzorowania obiektowo – relacyjnego. Są one powiązane z konkretnym językiem programowania

a) Hibernate – Najpopularniejsze narzędzie wykorzystywane dla języka Java. Jest on prosty w obsłudze oraz wydajny. Pozwala na zachowanie trwałości obiektów przy jednoczesnym wsparciu mechanizmów języka Java. Jest jedną z implementacji Java Persistence. Do sterowania można użyć zarówno XML jak i adnotacji.

b) Oracle TopLink – konkurencyjne w stosunku do Hibernate narzędzie dla języka Java, rozwijane przez firmę Oracle. Jego zaletą jest bardzo dobra integracja z innymi produktami Oracle takimi jak ich bazy danych.

c) Microsoft ADO.NET Entity Framework – framework dla języka C#. We wcześniejszych wersjach był częścią .NET, jednak od wersji 6 jest oddzielony.

W naszym projekcie skupimy się na wykorzystaniu Hibernate.

3. Metody sterowania:

Jeśli używamy Hibernate jako providera do JPA, to plikiem konfiguracyjnym jest `persistence.xml`, natomiast jeśli nie to plikiem konfiguracyjnym jest `hibernate.cfg.xml`.

Dla Hibernate można wykorzystać 2 metody sterowania: Pliki XML lub adnotacje.

a) Pliki XML

Na początku należy utworzyć klasę w Javie.

```
package com.testEntities;

import lombok.Getter;
import lombok.Setter;

public class Tank {

    @Setter
    @Getter
    private int id;

    @Getter
    @Setter
    private String name;
}
```

Powyższe adnotacje są powiązane z Lombokiem i służą jedynie utworzeniu odpowiednich getterów i setterów.

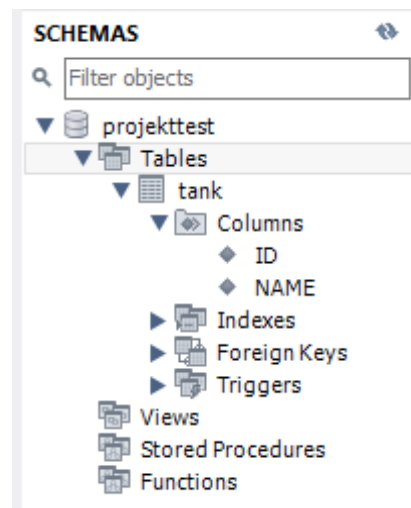
Następnie tworzymy plik `<nazwaKlasy>.hbm.xml`

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
    <class name = "com.testEntities.Tank" table="TANK">
        <id name="id" type="int" column="ID">
            <generator class="native"/>
        </id>
        <property name="name" column="NAME" type="string"/>
    </class>
</hibernate-mapping>
```

Jest to plik służący do mapowania danej klasy na bazę danych.

Po uruchomieniu Hibernate, w bazie danych zostanie utworzona odpowiednia tabela:



b) Adnotacje

Hibernate pozwala również na użycie adnotacji do mapowania klasy.

Wystarczy utworzyć klasę w Javie z odpowiednimi adnotacjami:

```
package com.testEntities;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

import lombok.Getter;
import lombok.Setter;

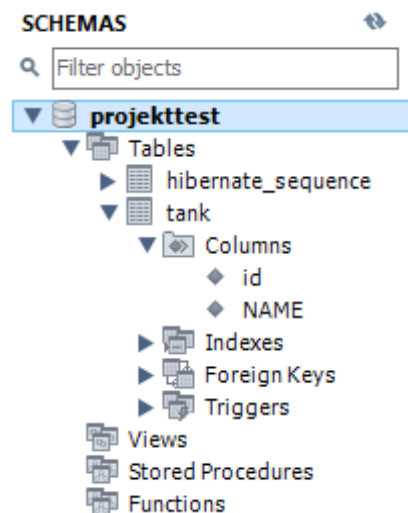
@Entity
@Table(name = "TANK") // W przypadku braku domyślnej nazwy tabeli jest taka jak klasy
public class Tank {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO) // Wartość kolumny jest
                                                    // generowana automatycznie

    @Getter
    private int id;

    @Getter
    @Setter
    @Column(name = "NAME") // Tak jak w przypadku @Table
    private String name;
}
```

Tabela utworzona po uruchomieniu Hibernate:



Największą zaletą adnotacji jest łatwość i szybkość pisania, a także przejrzystość kodu. Najlepiej używać ich kiedy trzeba stworzyć dużą ilość tabel na podstawie encji. Natomiast zaletą plików XML jest możliwość dokonywania zmian w bazie danych bez modyfikacji kodu aplikacji.

4. Sposoby odwzorowania dziedziczenia:

W Hibernate wyróżniamy 3 sposoby na odwzorowanie dziedziczenia: Single table per class hierarchy, Table per class hierarchy, Joined subclass.

a) Single table per class hierarchy:

Wszystkie klasy w hierarchii dziedziczenia są połączone w jedną tabelę w bazie. Powoduje to, że pola, których dana klasa nie posiada są nullami. Jest to najlepsze rozwiązanie dla nieskomplikowanych problemów.

Przykład:

Vehicle.java:

```
package com.testEntities;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Inheritance;
import javax.persistence.InheritanceType;

import lombok.Getter;
```

```
import lombok.Setter;

@Entity
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
public abstract class Vehicle {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Getter
    private int id;

    @Getter
    @Setter
    private String name;

}
```

Tank.java

```
package com.testEntities;

import javax.persistence.Entity;

import lombok.Getter;
import lombok.Setter;

@Entity
public class Tank extends Vehicle{

    @Getter
    @Setter
    private String gunModel;

}
```

Plane.java

```
package com.testEntities;

import javax.persistence.Entity;

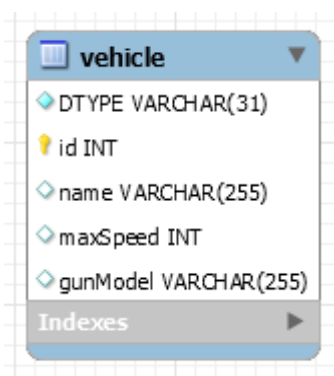
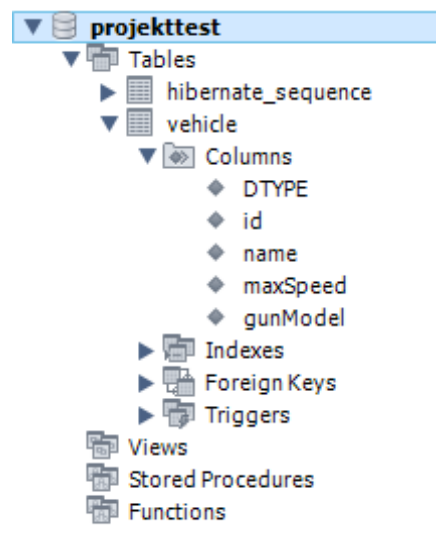
import lombok.Getter;
import lombok.Setter;

@Entity
public class Plane extends Vehicle{

    @Getter
    @Setter
    private int maxSpeed;

}
```

Po uruchomieniu Hibernate baza wygląda następująco:



b) Table per class hierarchy:

W tej strategii każda klasa w hierarchii otrzymuje swoją własną tabelę. Powoduje to, że informacje mogą być zdublowane. Najlepiej sprawdza się, gdy dane w bazie są prawie niezmiennie.

Przykład:

W stosunku do poprzedniego przykładu zmianie uległa tylko klasa **Vehicle.java**

```
package com.testEntities;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Inheritance;
import javax.persistence.InheritanceType;

import lombok.Getter;
import lombok.Setter;

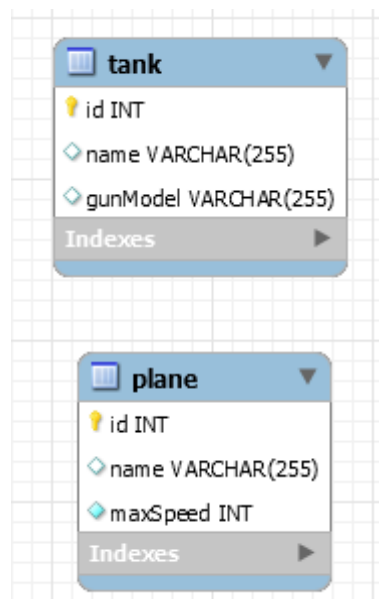
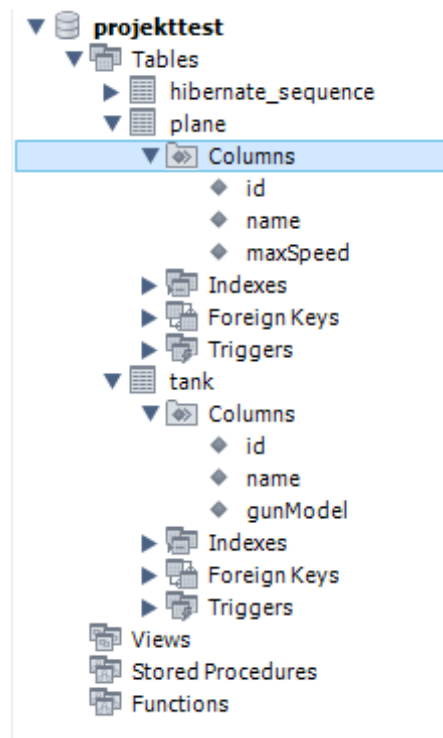
@Entity
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
public abstract class Vehicle {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Getter
    private int id;

    @Getter
    @Setter
    private String name;

}
```

Baza wygląda następująco:



c) Joined subclass

Jest to najwerniejsze odwzorowanie hierarchii, jednak utrudnia to wydobycie z bazy konkretnych informacji. Każda klasa jest mapowana na poprzednią w hierarchii. Strategia ta wykorzystuje relacje.

Przykład:

W stosunku do poprzedniego przykładu zmianie uległa tylko klasa **Vehicle.java**

```
package com.testEntities;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Inheritance;
import javax.persistence.InheritanceType;

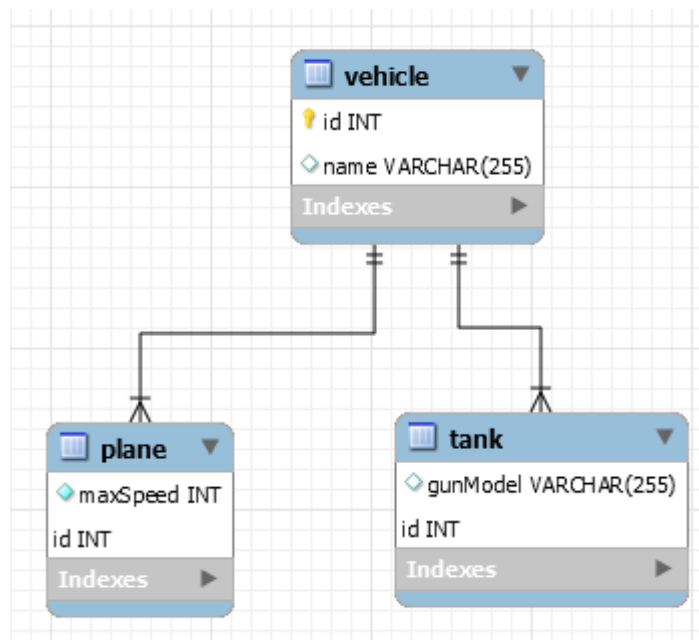
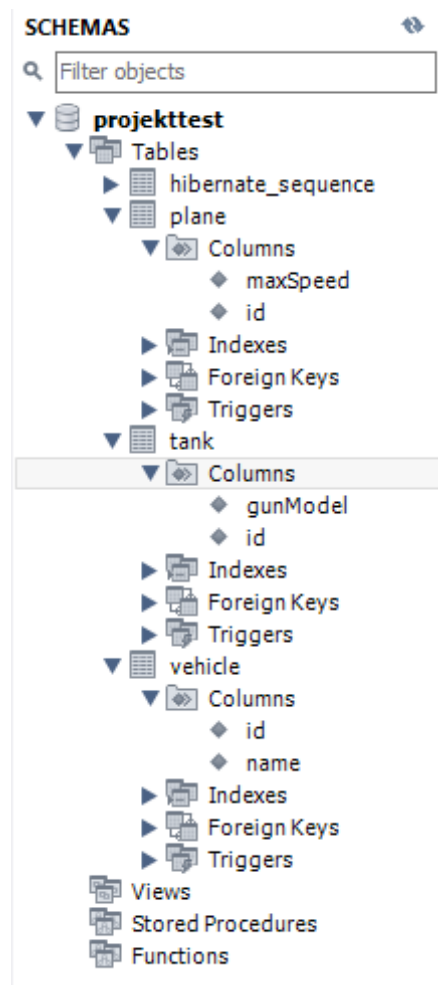
import lombok.Getter;
import lombok.Setter;

@Entity
@Inheritance(strategy = InheritanceType.JOINED)
public abstract class Vehicle {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Getter
    private int id;

    @Getter
    @Setter
    private String name;
}
```

Baza wygląda następująco:



5. Asocjacje:

Asocjacje występujące w Hibernate możemy podzielić następująco:

a) Ze względu na kierunek:

- *jednokierunkowe

- *dwukierunkowe

b) Ze względu na powiązanie:

- *one-to-one

- *many-to-many

- *one-to-many

- *many-to-one

Jednokierunkowe:

Nie wymagają one mapowania. Jedyne co trzeba zrobić to do encji dowiązać referencję na inną encję, oraz użyć odpowiedniej adnotacji. Używana kiedy związek encji 1 z encją 2 nie oznacza związku encji 2 z 1.

Dwukierunkowe:

Trzeba je dodatkowo zmapować. Jest to spowodowane tym, że Hibernate widzi wszystkie relacje jako jednokierunkowe.

One-to-one:

Odwzorowanie relacji 1 do 1. W przypadku relacji dwukierunkowej klucz obcy może znajdować się po dowolnej stronie.

Relacja jednokierunkowa:

Company.java:

```
package com.testEntities;
```

```
import javax.persistence.GeneratedValue;
```

```
import javax.persistence.GenerationType;
```

```
import javax.persistence.Id;
```

```
import lombok.Getter;
```

```
import lombok.Setter;
```

```

@Entity
public class Company {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Getter
    private long id;

    @Getter
    @Setter
    private int number;

}

```

Battalion.java:

```

package com.testEntities;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToOne;

import lombok.Getter;
import lombok.Setter;

@Entity
public class Battalion {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Getter
    private long id;

    @Getter
    @Setter
    private int number;

    @Getter
    @Setter
    private String type;

    @Getter
    @Setter

```

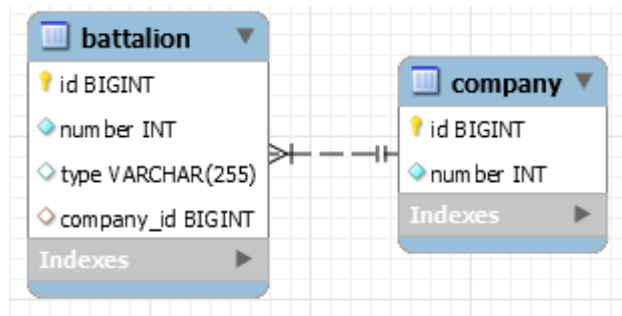
```

    @OneToOne
    private Company company;

}

```

Wygenerowana baza:



Relacja dwukierunkowa:

Relację dwukierunkową one-to-one można zaimplementować na 3 sposoby:

- a) Za pomocą klucza obcego
- b) Za pomocą dzielonego klucza głównego
- c) Za pomocą Join table

a)

Company.java:

```
package com.testEntities;
```

```

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToOne;

```

```

import lombok.Getter;
import lombok.Setter;

```

```

@Entity
public class Company {

```

```

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Getter

```

```

    private long id;

    @Getter
    @Setter
    private int number;

    @Getter
    @Setter
    @OneToOne(mappedBy = "company")
    private Battalion battalion;
}

```

Battalion.java:

```

package com.testEntities;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToOne;

import lombok.Getter;
import lombok.Setter;

@Entity
public class Battalion {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Getter
    private long id;

    @Getter
    @Setter
    private int number;

    @Getter
    @Setter
    private String type;

    @Getter
    @Setter
    @OneToOne

```



```

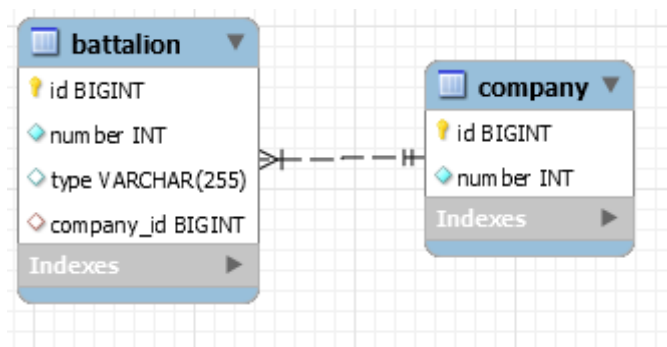
    @JoinColumn(referencedColumnName = "id")
    private Company company;

}

```

Jak widać aby stworzyć relację dwukierunkową trzeba było dodać referencję również do encji company, oraz odpowiednio je zmapować. Adnotacja `@JoinColumn` oznacza do której kolumny ma odnosić się Hibernate, aby utworzyć klucz obcy.

Baza po utworzeniu:



b)

Company.java:

```
package com.testEntities;
```

```

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.MapId;
import javax.persistence.OneToOne;

```

```

import lombok.Getter;
import lombok.Setter;

```

```

@Entity
public class Company {

```

```

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Getter
    private long id;

    @Getter
    @Setter

```

```

        private int number;

        @Getter
        @Setter
        @OneToOne(mappedBy = "company")
        private Battalion battalion;
    }

```

Battalion.java:

```

package com.testEntities;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.MapId;
import javax.persistence.OneToOne;

import lombok.Getter;
import lombok.Setter;

@Entity
public class Battalion {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Getter
    private long id;

    @Getter
    @Setter
    private int number;

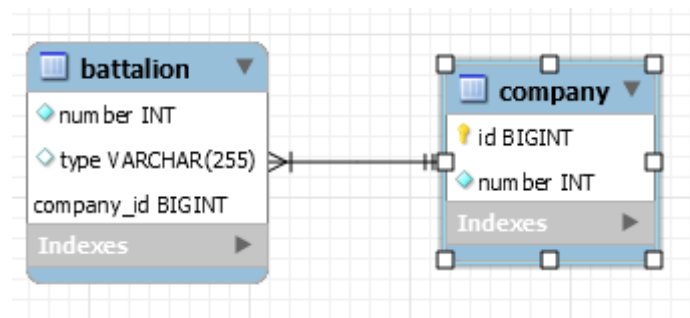
    @Getter
    @Setter
    private String type;

    @Getter
    @Setter
    @OneToOne
    @MapId
    private Company company;
}

```

Adnotacja @MapsId wskazuje, że należy użyć kolumny id danej encji jako klucza głównego i obcego.

Schemat bazy:



c)

Company.java:

```
package com.testEntities;
```

```
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
import javax.persistence.MapId;  
import javax.persistence.OneToOne;
```

```
import lombok.Getter;  
import lombok.Setter;
```

```
@Entity
```

```
public class Company {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.AUTO)
```

```
    @Getter
```

```
    private long id;
```

```
    @Getter
```

```
    @Setter
```

```
    private int number;
```

```
    @Getter
```

```
    @Setter
```

```
    @OneToOne(mappedBy = "company")
```

```
        private Battalion battalion;
    }
```

Battalion.java:

```
package com.testEntities;
```

```
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.MapId;
import javax.persistence.OneToOne;
```

```
import lombok.Getter;
import lombok.Setter;
```

```
@Entity
```

```
public class Battalion {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.AUTO)
```

```
    @Getter
```

```
    private long id;
```

```
    @Getter
```

```
    @Setter
```

```
    private int number;
```

```
    @Getter
```

```
    @Setter
```

```
    private String type;
```

```
    @Getter
```

```
    @Setter
```

```
    @OneToOne
```

```
    @JoinTable(name="battalion_company", joinColumns =
```

```
    @JoinColumn(name="id_Battalion"), inverseJoinColumns =
```

```
    @JoinColumn(name="id_Company"))
```

```
    private Company company;
```

```
}
```

Adnotacja @JoinTable oznacza, że należy utworzyć tabelę pośrednią utworzoną z 2 encji.

Schemat bazy:



Many-to-many:

Relacja jednokierunkowa:

Company.java:

```
package com.testEntities;
```

```
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;
```

```
import lombok.Getter;  
import lombok.Setter;
```

```
@Entity  
public class Company {
```

```
    @Id  
    @GeneratedValue(strategy = GenerationType.AUTO)  
    @Getter  
    private long id;
```

```
    @Getter  
    @Setter  
    private int number;
```

```
}
```

Battalion.java:

```
package com.testEntities;

import java.util.ArrayList;
import java.util.List;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToMany;

import lombok.Getter;
import lombok.Setter;

@Entity
public class Battalion {

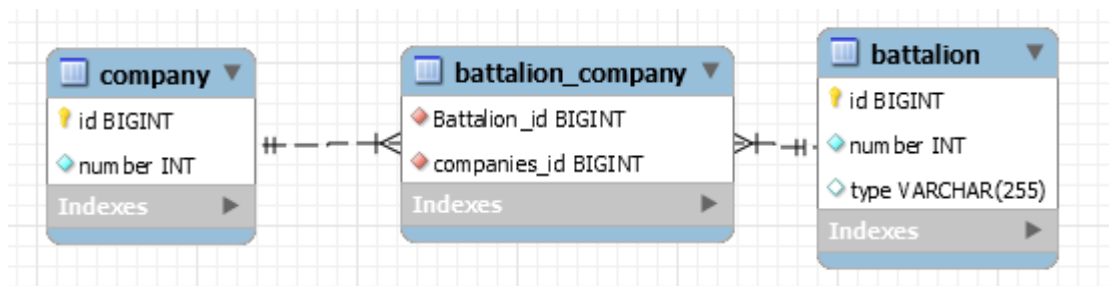
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Getter
    private long id;

    @Getter
    @Setter
    private int number;

    @Getter
    @Setter
    private String type;

    @Getter
    @Setter
    @ManyToMany
    List<Company> companies = new ArrayList<Company>();
}
```

Schemat bazy:



Relacja dwukierunkowa:

Company.java:

```
package com.testEntities;
```

```
import java.util.ArrayList;
```

```
import java.util.List;
```

```
import javax.persistence.Entity;
```

```
import javax.persistence.GeneratedValue;
```

```
import javax.persistence.GenerationType;
```

```
import javax.persistence.Id;
```

```
import javax.persistence.ManyToMany;
```

```
import lombok.Getter;
```

```
import lombok.Setter;
```

```
@Entity
```

```
public class Company {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.AUTO)
```

```
    @Getter
```

```
    private long id;
```

```
    @Getter
```

```
    @Setter
```

```
    private int number;
```

```
    @Getter
```

```
    @ManyToMany(mappedBy = "companies")
```

```
    List<Battalion> battalions=new ArrayList<Battalion>();
```

```
}
```

Battalion.java:

```
package com.testEntities;

import java.util.ArrayList;
import java.util.List;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;

import lombok.Getter;
import lombok.Setter;

@Entity
public class Battalion {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Getter
    private long id;

    @Getter
    @Setter
    private int number;

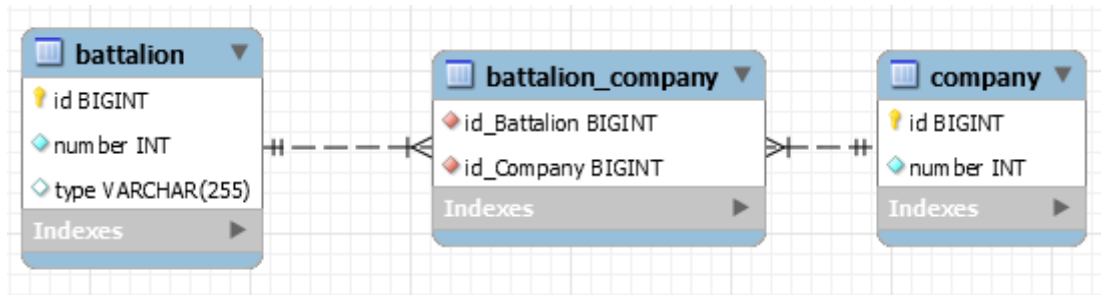
    @Getter
    @Setter
    private String type;

    @Getter
    @ManyToMany
    @JoinTable(name="battalion_company", joinColumns =
    @JoinColumn(name="id_Battalion"), inverseJoinColumns =
    @JoinColumn(name="id_Company"))
    List<Company> companies = new ArrayList<Company>();

}
```


Jak widać relację również należy odpowiednio zmapować.

Schemat bazy:



W następnej części omówimy relacje many-to-one oraz one-to-many. Ze względu na ich specyfikę w części poświęconej relacjom jednokierunkowym omówimy je osobno, natomiast w części dwukierunkowej zestawimy je.

Jednokierunkowa one-to-many:

Domyślnie Hibernate tworzy tabelę pośrednią, jednak można też zastosować klucz obcy.

Tabela pośrednia:

Company.java:

```
package com.testEntities;
```

```
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
```

```
import lombok.Getter;
import lombok.Setter;
```

```
@Entity
```

```
public class Company {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.AUTO)
```

```
    @Getter
```

```
    private long id;
```

```
    @Getter
```

```
    @Setter
```

```
        private int number;
    }
}
```

Battalion.java:

```
package com.testEntities;

import java.util.ArrayList;
import java.util.List;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;

import lombok.Getter;
import lombok.Setter;

@Entity
public class Battalion {

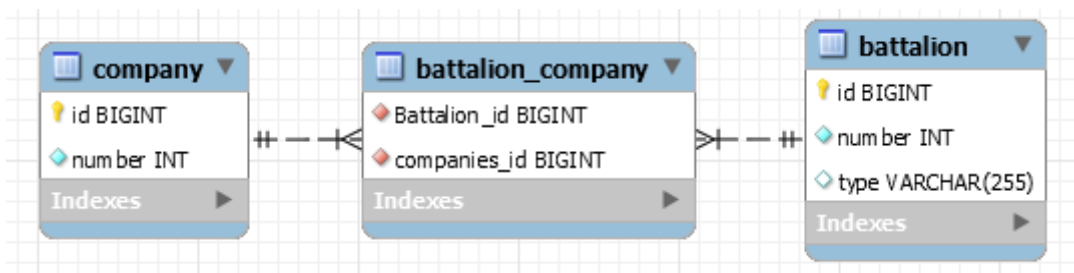
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Getter
    private long id;

    @Getter
    @Setter
    private int number;

    @Getter
    @Setter
    private String type;

    @Getter
    @OneToMany
    List<Company> companies = new ArrayList<Company>();
}
}
```

Utworzona baza:



Klucz obcy:

Jedyną modyfikacją kodu było dodanie w klasie Battalion.java adnotacji @JoinColumn.

```
package com.testEntities;
```

```
import java.util.ArrayList;
import java.util.List;
```

```
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToMany;
```

```
import lombok.Getter;
import lombok.Setter;
```

```
@Entity
```

```
public class Battalion {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.AUTO)
```

```
    @Getter
```

```
    private long id;
```

```
    @Getter
```

```
    @Setter
```

```
    private int number;
```

```
    @Getter
```

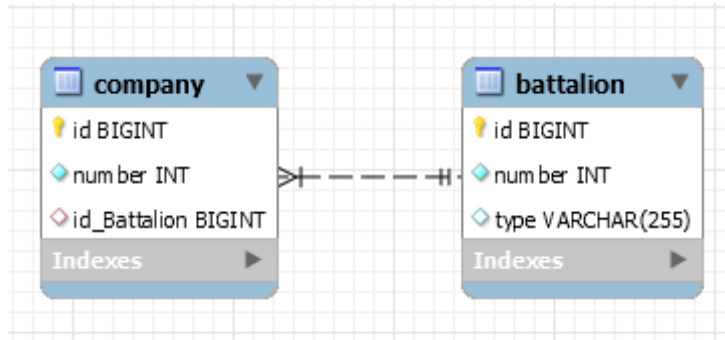
```
    @Setter
```

```
    private String type;
```

```

    @Getter
    @OneToMany
    @JoinColumn(name="id_Battalion")
    List<Company> companies = new ArrayList<Company>();
}

```



Jednokierunkowa many-to-one:

Company.java:

```

package com.testEntities;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;

import lombok.Getter;
import lombok.Setter;

@Entity
public class Company {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Getter
    private long id;

    @Getter
    @Setter
    private int number;

    @Getter
    @Setter

```

```

    @ManyToOne
    private Battalion battalion;
}

```

Battalion.java:

```
package com.testEntities;
```

```
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
```

```
import lombok.Getter;
import lombok.Setter;
```

```
@Entity
public class Battalion {
```

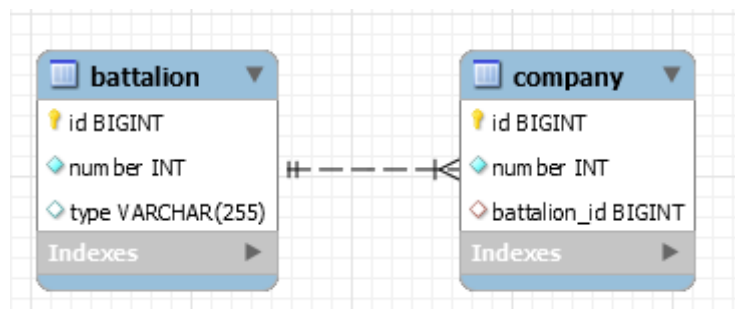
```
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Getter
    private long id;
```

```
    @Getter
    @Setter
    private int number;
```

```
    @Getter
    @Setter
    private String type;
```

```
}
```

Baza:



Jak widać przy zastosowaniu relacji many-to-one w Company efekt jest ten sam co przy zastosowaniu one-to-many z @JoinTable w Battalion.

Dwukierunkowa one-to-many i many-to-one:

W przypadku chęci stworzenia relacji dwukierunkowej musimy zastosować obie relacje i odpowiednio je zmapować.

Company.java:

```
package com.testEntities;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;

import lombok.Getter;
import lombok.Setter;

@Entity
public class Company {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Getter
    private long id;

    @Getter
    @Setter
    private int number;

    @Getter
    @Setter
    @ManyToOne
    private Battalion battalion;
}
```

Battalion.java:

```
package com.testEntities;

import java.util.ArrayList;
import java.util.List;
```

```

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;

import lombok.Getter;
import lombok.Setter;

@Entity
public class Battalion {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Getter
    private long id;

    @Getter
    @Setter
    private int number;

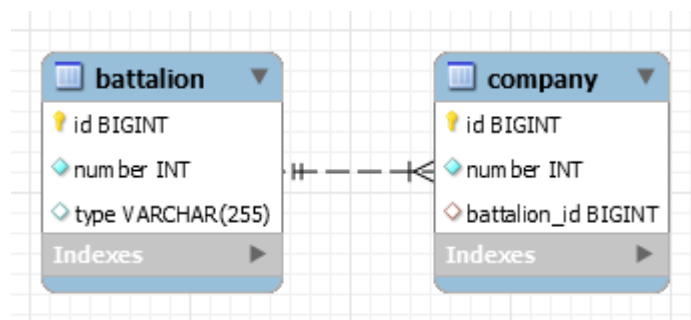
    @Getter
    @Setter
    private String type;

    @Getter
    @OneToMany(mappedBy = "battalion")
    private List<Company> companies= new ArrayList<Company>();
}

```

Tego typu relacje należy mapować w encji z adnotacją `@OneToMany`

Baza:



6. Aplikacja:

Nasza aplikacja odwzorowuje strukturę jednostki wojskowej od poziomu drużyny do batalionu piechoty.

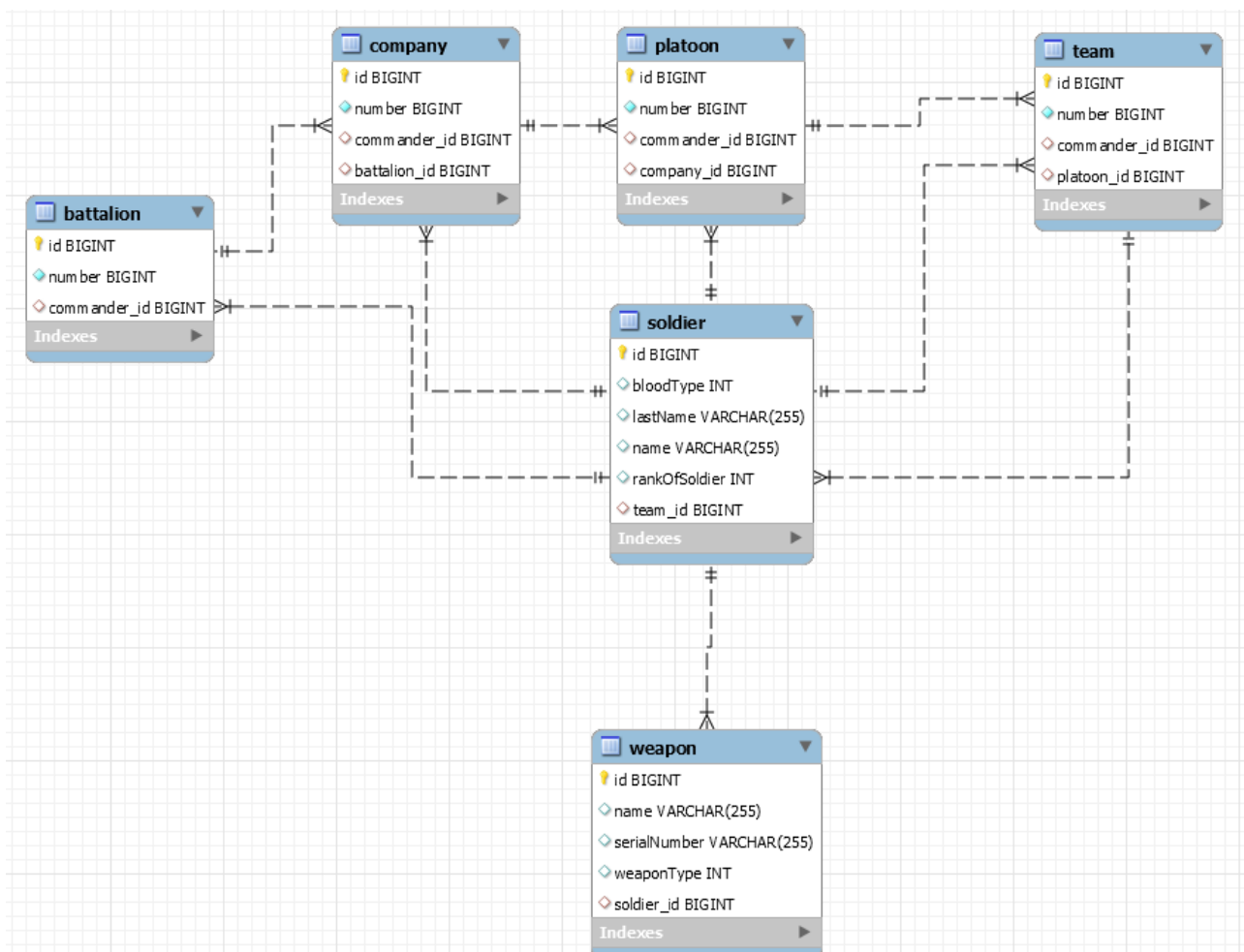
Użyte technologie:

- Java: Język programowania
- Eclipse: Środowisko programistyczne
- Hibernate: Odwzorowanie obiektowo – relacyjne
- MySQL: Baza danych
- Maven: Zapewnienie potrzebnych w projekcie bibliotek

Planowaliśmy również użycie Javy EE, JSP oraz serwera Tomcat. Po głębszej analizie problemu stwierdziliśmy jednak, że tworzenie aplikacji webowej nie jest konieczne, ze względu na tematykę projektu.

6.1) Baza danych

Schemat bazy danych stworzonej przy pomocy Hibernate wygląda następująco:



Zdecydowaliśmy się na odwzorowanie dziedziczenia Table per class hierarchy, ze względu na łatwość otrzymywania poszczególnych informacji z bazy oraz niewielką liczbę zmian danych w tabelach dziedziczących (struktura jednostki wojskowej nie zmienia się prawie wcale i zazwyczaj tylko na określony czas).

Listingi z kodami źródłowymi poszczególnych encji:

Unit.java:

```
package com.model.entities;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Inheritance;
import javax.persistence.InheritanceType;
import javax.persistence.OneToOne;

import lombok.Getter;
import lombok.Setter;
import lombok.ToString;

@Entity
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
public abstract class Unit implements Persistable{

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Getter
    private long id;

    @Getter
    @Setter
    private long number;

    @Getter
    @Setter
    @OneToOne
    private Soldier commander;
}
```

Battalion.java:

```
package com.model.entities;

import java.util.ArrayList;
import java.util.List;

import javax.persistence.Entity;
import javax.persistence.OneToMany;

import org.hibernate.annotations.LazyCollection;
import org.hibernate.annotations.LazyCollectionOption;
```

```

import lombok.Getter;
import lombok.ToString;

@Entity
@ToString(callSuper = true, exclude = "companies")
public class Battalion extends Unit implements Persistable {

    @Getter
    @OneToMany(mappedBy = "battalion")
    @LazyCollection(LazyCollectionOption.FALSE)
    private List<Company> companies = new ArrayList<Company>();

}

```

Company.java:

```

package com.model.entities;

import java.util.ArrayList;
import java.util.List;

import javax.persistence.Entity;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;

import org.hibernate.annotations.LazyCollection;
import org.hibernate.annotations.LazyCollectionOption;

import lombok.Getter;
import lombok.Setter;
import lombok.ToString;

@Entity
@ToString(callSuper = true, exclude = {"battalion", "platoons"})
public class Company extends Unit implements Persistable{

    @Getter
    @Setter
    @ManyToOne
    private Battalion battalion;

    @Getter
    @OneToMany(mappedBy = "company")
    @LazyCollection(LazyCollectionOption.FALSE)
    private List<Platoon> platoons = new ArrayList<Platoon>();

}

```

Platoon.java:

```

package com.model.entities;

import java.util.ArrayList;
import java.util.List;

import javax.persistence.Entity;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;

```

```

import org.hibernate.annotations.LazyCollection;
import org.hibernate.annotations.LazyCollectionOption;

import lombok.Getter;
import lombok.Setter;
import lombok.ToString;

@ToString(callSuper = true, exclude = {"company", "teams"})
@Entity
public class Platoon extends Unit implements Persistable{

    @Getter
    @Setter
    @ManyToOne
    private Company company;

    @Getter
    @OneToMany(mappedBy = "platoon")
    @LazyCollection(LazyCollectionOption.FALSE)
    private List<Team> teams = new ArrayList<Team>();
}

```

Team.java:

```

package com.model.entities;

import java.util.ArrayList;
import java.util.List;

import javax.persistence.Entity;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;

import org.hibernate.annotations.LazyCollection;
import org.hibernate.annotations.LazyCollectionOption;

import lombok.Getter;
import lombok.Setter;
import lombok.ToString;

@ToString(callSuper = true, exclude = {"platoon", "soldiers"})
@Entity
public class Team extends Unit implements Persistable{

    @Getter
    @Setter
    @ManyToOne
    private Platoon platoon;

    @Getter
    @OneToMany(mappedBy = "team")
    @LazyCollection(LazyCollectionOption.FALSE)
    private List<Soldier> soldiers = new ArrayList<Soldier>();
}

```

Soldier.java:

```
package com.model.entities;

import java.util.ArrayList;
import java.util.List;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;

import org.hibernate.annotations.LazyCollection;
import org.hibernate.annotations.LazyCollectionOption;

import com.utils.enums.BloodType;
import com.utils.enums.Rank;

import lombok.Getter;
import lombok.Setter;
import lombok.ToString;

@Entity
@ToString(exclude = {"team", "weapons"})
public class Soldier implements Persistable{

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Getter
    private long id;

    @Getter
    @Setter
    private String name;

    @Getter
    @Setter
    private String lastName;

    @Getter
    @Setter
    private BloodType bloodType;

    @Getter
    @Setter
    @Column(name = "rankOfSoldier")
    private Rank rank;

    @Getter
    @OneToMany(mappedBy = "soldier")
    @LazyCollection(LazyCollectionOption.FALSE)
    private List<Weapon> weapons = new ArrayList<Weapon>();

    @Getter
    @Setter
    @ManyToOne
    private Team team;
```

```

        public String getFullName(){
            return getName() + " " + getLastName();
        }
    }
}

```

Weapon.java:

```

package com.model.entities;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;

import com.utils.enums.WeaponType;

import lombok.Getter;
import lombok.Setter;
import lombok.ToString;

@ToString(exclude = "soldier")
@Entity
public class Weapon implements Persistable{

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Getter
    private long id;

    @Getter
    @Setter
    private String name;

    @Getter
    @Setter
    private String serialNumber;

    @Getter
    @Setter
    private WeaponType weaponType;

    @Getter
    @Setter
    @ManyToOne
    private Soldier soldier;
}

```

Adnotacja `@LazyCollection(LazyCollectionOption.FALSE)`, wyłącza domyślne leniwe kolekcje, czyli ładowanie obiektów-dzieci na żądanie (domyślnie włączone w od Hibernate 3.0) zastępując je ładowaniem ich od razu. Powoduje to rozwiązanie problemu z fetchowaniem, jednak zmniejsza szybkość działania, oraz może powodować wielokrotne wyszukanie tych samych danych (co jednak można bardzo prosto rozwiązać).

6.2) Aplikacja - wstęp

Nasza aplikacja odwzorowuje architekturę MVC (Model, View, Controller), gdzie poszczególne warstwy odpowiadają za inną funkcjonalność:

- Model – Komunikacja i odwzorowanie bazy danych – Encje, repozytoria do encji
- View – Interfejs użytkownika – Konsola
- Controller – Logika biznesowa aplikacji – Akcje

Funkcjonalności zaimplementowane w aplikacji:

- CRUD dla wszystkich encji w bazie danych
- Możliwość przypinania i odpinania encji do encji
- Różne sposoby wyszukiwania encji (po Id, w formie zapytania, itp.)
- Walidacja przy wykonywaniu akcji (np. jeśli żołnierz ma za niski stopień to nie może zostać dowódcą na Batalionu)

6.3) Akcje

Akcje ogólne:

Nazwa	Zadanie	Opis działania Algorytm
(Interfejs) Action	Stworzenie wspólnego interfejsu dla wszystkich akcji	Interfejs udostępnia dwie bezparametrowe metody. -Launch - w tej metodzie zawarta jest logika wykonywana przez akcję. -getName - służy do uzyskania nazwy identyfikującej daną akcję. Nazwy te są następnie wyświetlane na liście wyboru.
AssignCommanderToBattalion	Dołączenie Dowódcy do Batalionu	1. Wyszukanie docelowego Batalionu 2. Wyszukanie Żołnierza o kwalifikacjach dowódcy Batalionu 3. Usunięcie Żołnierza ze wszystkich dotychczasowych struktur aby zachować spójność danych 4. Dodanie Żołnierza jako dowódcę Batalionu 5. Aktualizacja Batalionu w bazie danych
AssignCommanderToCompany	Dołączenie Dowódcy do Kompanii	1. Wyszukanie docelowej Kompanii 2. Wyszukanie Żołnierza o kwalifikacjach dowódcy Kompanii 3. Usunięcie Żołnierza ze wszystkich dotychczasowych struktur aby zachować spójność danych 4. Dodanie Żołnierza jako dowódcę Kompanii 5. Aktualizacja Kompanii w bazie danych

AssignCommanderToPlatoon	Dołączenie Dowódcy do Plutonu	<ol style="list-style-type: none"> 1. Wyszukanie docelowego Plutonu 2. Wyszukanie Żołnierza o kwalifikacjach dowódcy Plutonu 3. Usunięcie Żołnierza ze wszystkich dotychczasowych struktur aby zachować spójność danych 4. Dodanie Żołnierza jako dowódcę Plutonu 5. Aktualizacja Plutonu w bazie danych
AssignCommanderToTeam	Dołączenie Dowódcy do Grupy	<ol style="list-style-type: none"> 1. Wyszukanie docelowej Grupy 2. Wyszukanie Żołnierza o kwalifikacjach dowódcy Grupy 3. Usunięcie Żołnierza ze wszystkich dotychczasowych struktur aby zachować spójność danych 4. Dodanie Żołnierza jako członka grupy 5. Dodanie Żołnierza jako dowódcę Grupy 6. Aktualizacja Grupy w bazie danych 7. Aktualizacja Żołnierza w bazie danych
AssignCompanyToBattalion	Dołączenie Kompanii do Batalionu	<ol style="list-style-type: none"> 1. Wyszukanie Batalionu 2. Wyszukanie Kompanii 3. Dodanie Kompanii do Batalionu o ile już w nim nie jest <ol style="list-style-type: none"> 3.1 Aktualizacja Batalionu w bazie danych 3.2 Aktualizacja Kompanii w bazie danych
AssignPlatoonToCompany	Dołączenie Plutonu do Kompanii	<ol style="list-style-type: none"> 1. Wyszukanie Kompanii 2. Wyszukanie Plutonu 3. Dodanie Plutonu do Kompanii o ile już w niej nie jest <ol style="list-style-type: none"> 3.1 Aktualizacja Kompanii w bazie danych 3.2 Aktualizacja Plutonu w bazie danych
AssignSoldierToTeam	Dodanie Żołnierza do Grupy	<ol style="list-style-type: none"> 1. Wyszukanie Grupy 2. Wyszukanie Żołnierza 3. Dodanie Żołnierza do grupy 4. Dodanie Żołnierza do grupy 5. Aktualizacja Grupy w bazie danych 6. Aktualizacja Żołnierza w bazie danych
AssignTeamToPlatoon	Dodanie Grupy do Plutonu	<ol style="list-style-type: none"> 1. Wyszukanie Plutonu 2. Wyszukanie Grupy 3. Dodanie Grupy do plutonu o ile w nim nie jest

		3.1 Aktualizacja Plutonu w bazie danych 3.2 Aktualizacja Grupy w bazie danych
AssignWeaponToSoldier	Przydzielenie nowej broni do wyposażenia Żołnierza	1. Wyszukanie Żołnierza 2. Wyszukanie broni 3. Przypisanie żołnierza do konkretnej broni 4. Dodanie broni do kolekcji Żołnierza 5. Aktualizacja Żołnierza w bazie danych 6. Aktualizacja Broni w bazie danych
Exit	Zakończenie działania aplikacji	1. Wywołanie funkcji systemowej służącej do przerywania działania aplikacji

Akcje Batalionu:

Nazwa	Zadanie	Opis działania Algorytm
CreateBattalion	Stworzenie nowego Batalionu	1. Stworzenie nowej instancji Batalionu 2. Pobranie od użytkownika identyfikatora 3. Przypisanie identyfikatora do Batalionu 4. Zapisanie Batalionu w bazie danych
DeleteBattalion	Usunięcie z bazy danych wskazanego Batalionu	1. Wyszukanie Batalionu 2. Odłączenie Dowódcy od Batalionu 3. Odłączenie Kompanii od Batalionu 4. Usunięcie Batalionu w bazie danych
FindAllBattalions	Wyszukanie wszystkich dostępnych Batalionów	1. Wyszukanie w bazie Batalionów 2. Wyświetlenie użytkownikowi listy wyników
FindBattalionByCommander	Wyszukanie Batalionu ze wskazanym Dowódcą	1. Pobranie danych Dowódcy 2. Wyszukanie listy Żołnierzy o pasujących danych 3. Dla każdego Dowódcy pobranie jego Batalionu 3.1 Wyświetlenie listy Batalionów danego Dowódcy
FindBattalionById	Wyszukanie Batalionu po numerze ID	1. Wyszukanie Batalionu 2. Wyświetlenie Batalionu 3. Wyświetlenie Kompanii należących do Batalionu
FindBattalionsWithoutCommander	Wyszukanie Batalionów które nie	1. Wyszukanie Batalionów bez Dowódcy w bazie danych

	mają Dowódcy	2. Wyświetlenie listy wyników użytkownikowi
FindBattalionsWithoutCompanies	Wyszukanie Batalionów które nie mają żadnych Kompanii	1. Wyszukanie Batalionów bez Kompanii w bazie danych 2. Wyświetlenie listy wyników użytkownikowi
RemoveCommanderFromBattalion	Usunięcie Dowódcy z wybranego Batalionu	1. Wyszukanie Batalionu 2. Ustawienie Dowódcy jako nie zdefiniowany (null) 3. Aktualizacja Batalionu w bazie danych
UpdateBattalion	Aktualizacja Batalionu	1. Wyszukanie Batalionu 2. Pobranie nowego numeru identyfikującego 3. Ustawienie nowego identyfikatora 4. Aktualizacja Batalionu w bazie danych

Akcje Kompanii:

Nazwa	Zadanie	Opis działania Algorytm
CreateCompany	Stworzenie nowej Kompanii	1. Stworzenie nowej instancji Kompanii 2. Pobranie od użytkownika identyfikatora 3. Przypisanie identyfikatora do Kompanii 4. Zapisanie Kompanii w bazie danych
DeleteCompany	Usunięcie z bazy danych wskazanej Kompanii	1. Wyszukanie Kompanii 2. Odłączenie Dowódcy od Kompanii 3. Odłączenie Plutonów od Kompanii 4. Usunięcie Kompanii w bazie danych
FindAllCompanies	Wyszukanie wszystkich dostępnych Kompanii	1. Wyszukanie w bazie Kompanii 2. Wyświetlenie użytkownikowi listy wyników
FindCompaniesWithoutBattalion	Wyszukanie Kompanii które nie mają Batalionu	1. Wyszukanie w bazie Kompanii 2. Wyświetlenie użytkownikowi listy wyników
FindCompaniesWithoutCommander	Wyszukanie Kompanii które nie mają dowódcy	1. Wyszukanie w bazie Kompanii 2. Wyświetlenie użytkownikowi listy wyników
FindCompaniesWithoutPlatoons	Wyszukanie Kompanii które nie mają Plutonów	1. Wyszukanie w bazie Kompanii 2. Wyświetlenie użytkownikowi listy wyników
FindCompanyByCommander	Wyszukanie Kompanii ze wskazanym Dowódcą	1. Pobranie danych Dowódcy 2. Wyszukanie listy Żołnierzy o pasujących danych 3. Dla każdego Dowódcy pobranie jego Kompanii 3.1 Wyświetlenie listy Kompanii danego Dowódcy
FindCompanyById	Wyszukanie Kompanii po numerze ID	1. Wyszukanie Kompanii 2. Wyświetlenie Kompanii 3. Wyświetlenie Batalionu do którego należy kompania 3. Wyświetlenie Plutonów należących do Batalionu
RemoveCommanderFromCompany	Usunięcie Dowódcy z wybranej Kompanii	1. Wyszukanie Kompanii 2. Ustawienie Dowódcy jako nie zdefiniowany (null) 3. Aktualizacja Kompanii w bazie danych
UpdateCompany	Aktualizacja Kompanii	1. Wyszukanie Kompanii 2. Pobranie nowego numeru identyfikującego 3. Ustawienie nowego identyfikatora 4. Aktualizacja Kompanii w bazie danych

Akcje Plutonu:

Nazwa	Zadanie	Opis działania Algorytm
CreatePlatoon	Stworzenie nowego Plutonu	1. Stworzenie nowej instancji Plutonu 2. Pobranie od użytkownika identyfikatora 3. Przypisanie identyfikatora do Plutonu 4. Zapisanie Plutonu w bazie danych
DeletePlatoon	Usunięcie z bazy danych wskazanego Plutonu	1. Wyszukanie Plutonu 2. Odłączenie Dowódcy od Plutonu 3. Odłączenie Grup od Plutonu 4. Usunięcie Plutonu w bazie danych
FindAllPlatoons	Wyszukanie wszystkich dostępnych Plutonów	1. Wyszukanie w bazie Plutonów 2. Wyświetlenie użytkownikowi listy wyników
FindPlatoonByCommander	Wyszukanie Plutonu ze wskazanym Dowódcą	1. Pobranie danych Dowódcy 2. Wyszukanie listy Żołnierzy o pasujących danych 3. Dla każdego Dowódcy pobranie jego Plutonów 3.1 Wyświetlenie listy Plutonów danego Dowódcy
FindPlatoonById	Wyszukanie Plutonu po numerze ID	1. Wyszukanie Plutonu 2. Wyświetlenie Plutonu 3. Wyświetlenie Kompanii danego Plutonu 4. Wyświetlenie listy Grup danego Plutonu
FindPlatoonsWithoutCommander	Wyszukanie Plutonów które nie mają Dowódcy	1. Wyszukanie Plutonów bez Dowódcy w bazie danych 2. Wyświetlenie listy wyników użytkownikowi
FindPlatoonsWithoutCompanies	Wyszukanie Plutonów które nie mają Kompanii	1. Wyszukanie Plutonów bez Kompanii w bazie danych 2. Wyświetlenie listy wyników użytkownikowi
FindPlatoonsWithoutTeams	Wyszukanie Plutonów które nie mają żadnych Grup	1. Wyszukanie Plutonów bez Grup w bazie danych 2. Wyświetlenie listy wyników użytkownikowi
RemoveCommanderFromPlatoon	Usunięcie Dowódcy z wybranego Plutonu	1. Wyszukanie Plutonu 2. Ustawienie Dowódcy jako nie zdefiniowany (null) 3. Aktualizacja Plutonu w bazie danych
UpdatePlatoon	Aktualizacja Plutonu	1. Wyszukanie Plutonu 2. Pobranie nowego numeru identyfikującego 3. Ustawienie nowego identyfikatora

		4. Aktualizacja Plutonu w bazie danych
--	--	--

Akcje Żołnierza:

Nazwa	Zadanie	Opis działania Algorytm
CreateSoldier	Stworzenie nowego żołnierza	<ol style="list-style-type: none"> 1. Stworzenie nowej instancji żołnierza 2. Pobranie od użytkownika imienia i przypisanie go do żołnierza 3. Pobranie od użytkownika nazwiska i przypisanie go do żołnierza 4. Pobranie od użytkownika poprawnej grupy krwi i przypisanie jej do żołnierza 5. Pobranie od użytkownika poprawnego stopnia i przypisanie go do żołnierza 6. Zapisanie żołnierza w bazie danych
DeleteSoldier	Usunięcie z bazy danych wskazanego żołnierza	<ol style="list-style-type: none"> 1. Wyszukanie żołnierza 2. Odłączenie broni od żołnierza 3. Odłączenie żołnierza od drużyny 4. Odłączenie dowódcy od drużyny 5. Odłączenie dowódcy od plutonu 6. Odłączenie dowódcy od kompanii 7. Odłączenie dowódcy od batalionu 8. Usunięcie żołnierza z bazy danych
FindAllSoldiers	Wyszukanie wszystkich dostępnych w bazie żołnierzy	<ol style="list-style-type: none"> 1. Wyszukanie w bazie żołnierzy 2. Wyświetlenie listy wyników
FindSoldierById	Wyszukanie żołnierza po ID	<ol style="list-style-type: none"> 1. Wyszukanie żołnierza 2. Wyświetlenie żołnierza 3. Wyświetlenie drużyny danego żołnierza
FindSoldiersWithoutTeam	Wyszukanie żołnierzy bez drużyn	<ol style="list-style-type: none"> 1. Wyszukanie żołnierzy bez drużyn w bazie danych 2. Wyświetlenie listy wyników
FindSoldiersWithoutWeapon	Wyszukanie żołnierzy bez broni	<ol style="list-style-type: none"> 1. Wyszukanie żołnierzy bez broni w bazie danych 2. Wyświetlenie listy wyników
UpdateSoldier	Aktualizacja żołnierza	<ol style="list-style-type: none"> 1. Wyszukanie żołnierza 2. Pobranie nowego imienia 3. Ustawienie nowego imienia 4. Pobranie nowego nazwiska

		5. Ustawienie nowego nazwiska 6. Pobranie nowej grupy krwi 7. Ustawienie nowej grupy krwi 8. Pobranie nowego stopnia 9. Ustawienie nowego stopnia 10. Aktualizacja żołnierza w bazie danych
--	--	--

Akcje drużyny:

Nazwa	Zadanie	Opis działania Algorytm
CreateTeam	Stworzenie nowej drużyny	1. Stworzenie nowej instancji drużyny 2. Pobranie od użytkownika numeru i przypisanie go do drużyny 3. Zapisanie drużyny w bazie danych
DeleteTeam	Usunięcie z bazy danych wskazanej drużyny	1. Wyszukanie drużyny 2. Odłączenie dowódcy od drużyny 3. Odłączenie żołnierzy od drużyny 4. usunięcie drużyny z bazy danych
FindAllTeams	Wyszukanie w bazie wszystkich dostępnych drużyn	1. Wyszukanie w bazie drużyn 2. Wyświetlenie listy wyników
FindTeamByCommander	Wyszukanie drużyny ze wskazanym dowódcą	1. Pobranie danych dowódcy 2. Wyszukanie listy żołnierzy o pasujących danych 3. Dla każdego dowódcy pobranie jego drużyn 4. Wyświetlenie listy drużyn danego dowódcy
FindTeamById	Wyszukanie drużyny po ID	1. Wyszukanie drużyny 2. Wyświetlenie drużyny 3. Wyświetlenie żołnierzy w drużynie
FindTeamsWithoutCommander	Wyszukanie drużyn bez dowódców	1. Wyszukanie drużyn bez dowódców w bazie danych 2. Wyświetlenie listy wyników
FindTeamsWithoutPlatoon	Wyszukanie drużyn bez plutonów	1. Wyszukanie drużyn bez plutonów w bazie danych 2. Wyświetlenie listy wyników
FindTeamsWithoutSoldiers	Wyszukanie drużyn bez żołnierzy	1. Wyszukanie drużyn bez żołnierzy w bazie danych 2. wyświetlenie listy wyników
RemoveCommanderFromTeam	Usunięcie dowódcy z danej drużyny	1. Wyszukanie danej drużyny 2. Ustawienie dowódcy w drużynie na null 3. Aktualizacja drużyny w bazie
UpdateTeam	Aktualizacja drużyny	1. Wyszukanie drużyny 2. Ustawienie nowego numeru drużyny

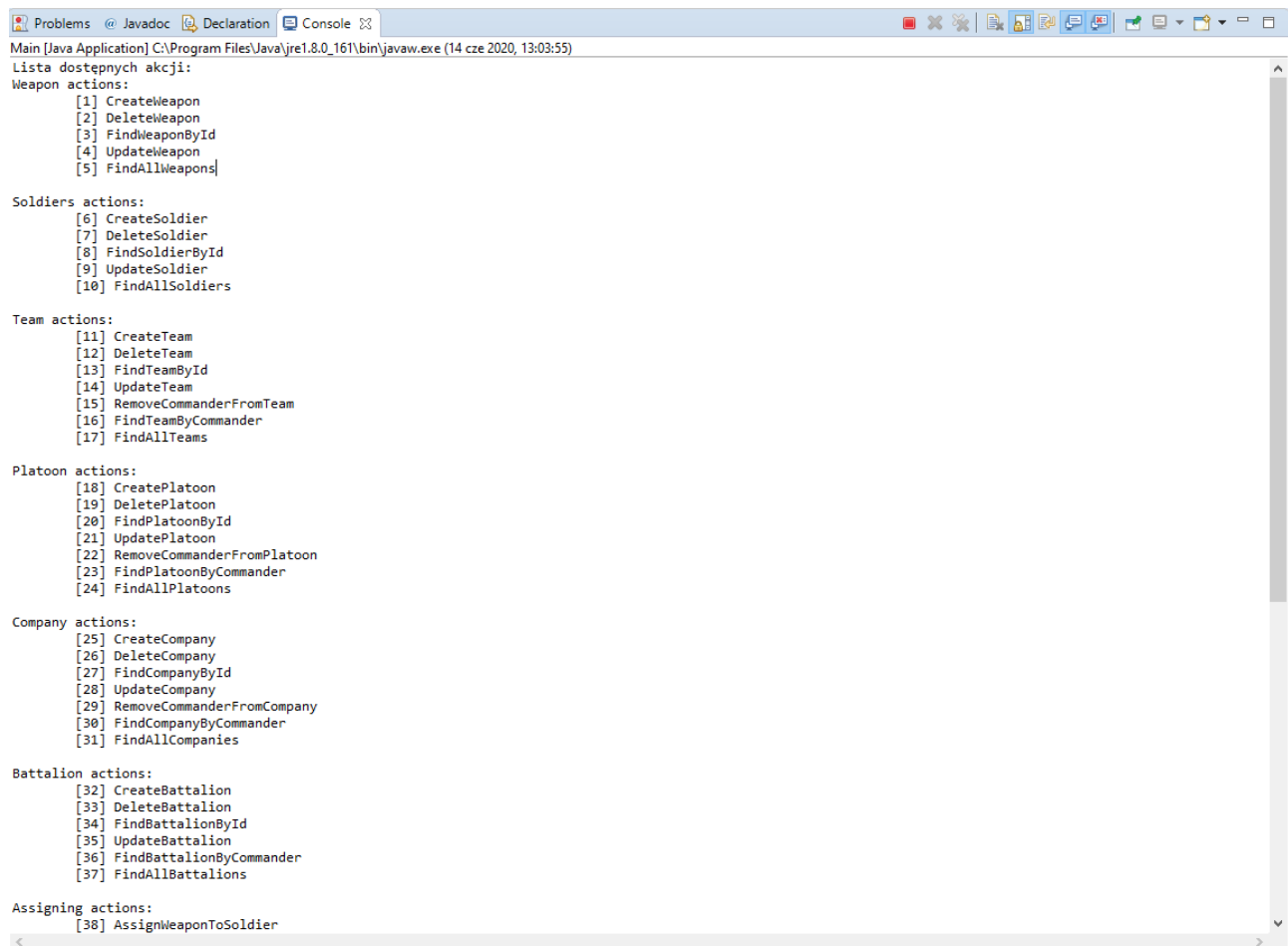
		3. Aktualizacja drużyny w bazie
--	--	---------------------------------

Akcje broni:

Nazwa	Zadanie	Opis działania Algorytm
CreateWeapon	Stworzenie nowej broni	<ol style="list-style-type: none"> 1. Stworzenie nowej instancji broni 2. Pobranie od użytkownika nazwy i przypisanie jej do broni 3. Pobranie od użytkownika numeru seryjnego i przypisanie go do broni 4. Pobranie do użytkownika poprawnego typu i przypisanie go do broni 5. Zapisanie broni w bazie danych
DeleteWeapon	Usunięcie wybranej broni	<ol style="list-style-type: none"> 1. Wyszukanie broni 2. Usunięcie broni z bazy
FindAllWeapons	Wyszukanie całej dostępnej broni z bazy	<ol style="list-style-type: none"> 1. Wyszukanie w bazie broni 2. Wyświetlenie listy wyników
FindWeaponById	Wyszukanie w bazie broni o podanym ID	<ol style="list-style-type: none"> 1. Wyszukanie broni 2. Wyświetlenie broni 3. Wyświetlenie żołnierza przypisanego do broni
FindWeaponWithoutSoldier	Wyszukanie broni bez żołnierzy	<ol style="list-style-type: none"> 1. Wyszukanie broni bez żołnierzy w bazie adnych 2. Wyświetlenie listy wyników
UpdateWeapon	Aktualizacja broni	<ol style="list-style-type: none"> 1. Wyszukanie broni 2. Pobranie nowej nazwy 3. Ustawienie nowej nazwy 4. Pobranie nowego numeru seryjnego 5. Ustawienie nowego numeru seryjnego 6. Pobranie nowego typu 7. Ustawienie nowego typu 8. Aktualizacja broni w bazie

6.3) Działanie aplikacji

W poniższym dziale przedstawimy zrzuty ekranów z działania wybranych akcji.



```

Main [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (14 cze 2020, 13:03:55)
Lista dostępnych akcji:
Weapon actions:
  [1] CreateWeapon
  [2] DeleteWeapon
  [3] FindWeaponById
  [4] UpdateWeapon
  [5] FindAllWeapons

Soldiers actions:
  [6] CreateSoldier
  [7] DeleteSoldier
  [8] FindSoldierById
  [9] UpdateSoldier
  [10] FindAllSoldiers

Team actions:
  [11] CreateTeam
  [12] DeleteTeam
  [13] FindTeamById
  [14] UpdateTeam
  [15] RemoveCommanderFromTeam
  [16] FindTeamByCommander
  [17] FindAllTeams

Platoon actions:
  [18] CreatePlatoon
  [19] DeletePlatoon
  [20] FindPlatoonById
  [21] UpdatePlatoon
  [22] RemoveCommanderFromPlatoon
  [23] FindPlatoonByCommander
  [24] FindAllPlatoons

Company actions:
  [25] CreateCompany
  [26] DeleteCompany
  [27] FindCompanyById
  [28] UpdateCompany
  [29] RemoveCommanderFromCompany
  [30] FindCompanyByCommander
  [31] FindAllCompanies

Battalion actions:
  [32] CreateBattalion
  [33] DeleteBattalion
  [34] FindBattalionById
  [35] UpdateBattalion
  [36] FindBattalionByCommander
  [37] FindAllBattalions

Assigning actions:
  [38] AssignWeaponToSoldier

```

Lista akcji

```
Problems Javadoc Declaration Console
Main [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (14 cze 2020, 13:03:55)

[32] CreateBattalion
[33] DeleteBattalion
[34] FindBattalionById
[35] UpdateBattalion
[36] FindBattalionByCommander
[37] FindAllBattalions

Assigning actions:
[38] AssignWeaponToSoldier
[39] AssignSoldierToTeam
[40] AssignCommanderToTeam
[41] AssignTeamToPlatoon
[42] AssignCommanderToPlatoon
[43] AssignPlatoonToCompany
[44] AssignCommanderToCompany
[45] AssignCompanyToBattalion
[46] AssignCommanderToBattalion

Advanced finding actions:
[47] FindWeaponsWithoutSoldier
[48] FindSoldiersWithoutTeam
[49] FindSoldiersWithoutWeapon
[50] FindTeamsWithoutSoldiers
[51] FindTeamsWithoutPlatoon
[52] FindTeamsWithoutCommander
[53] FindPlatoonsWithoutTeams
[54] FindPlatoonsWithoutCompany
[55] FindPlatoonsWithoutCommander
[56] FindCompaniesWithoutPlatoons
[57] FindCompaniesWithoutBattalion
[58] FindCompaniesWithoutCommander
[59] FindBattalionsWithoutCompanies
[60] FindBattalionsWithoutCommander

System actions:
[61] Exit

Podaj akcje
CreateWeapon
Podaj nazwę
AKS-74
Podaj nr. seryjny
AS321
Dostępne typy broni
0:ASSAULT_RIFLE
1:LMG
2:RPG
3:SNIPER_RIFLE
4:PISTOL
Podaj typ broni
ASSAULT_RIFLE
```

Akcja: Create Weapon

```
Problems Javadoc Declaration Console
Main [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (14 cze 2020, 13:03:55)

WARN: HHH10001002: Using Hibernate built-in connection pool (not for production use)
cze 14, 2020 1:06:13 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001005: using driver [com.mysql.cj.jdbc.Driver] at URL [jdbc:mysql://localhost:3306/techobdb?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC]
cze 14, 2020 1:06:13 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001001: Connection properties: {user=ToAdmin, password=****}
cze 14, 2020 1:06:13 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001003: Autocommit mode: false
cze 14, 2020 1:06:13 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PooledConnections <init>
INFO: HHH1000115: Hibernate connection pool size: 20 (min=1)
cze 14, 2020 1:06:13 PM org.hibernate.dialect.Dialect <init>
INFO: HHH1000400: Using dialect: org.hibernate.dialect.MySQL8Dialect
cze 14, 2020 1:06:14 PM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolatedConnection
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnectionAccess]
cze 14, 2020 1:06:14 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH1000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
Weapon(id=64, name=k, serialNumber=k, weaponType=ASSAULT_RIFLE)
Weapon(id=86, name=www, serialNumber=gghh, weaponType=RPG)
Weapon(id=103, name=PKM, serialNumber=1234, weaponType=LMG)
Weapon(id=105, name=AKS-74, serialNumber=AS321, weaponType=ASSAULT_RIFLE)

Lista dostępnych akcji:
Weapon actions:
[1] CreateWeapon
[2] DeleteWeapon
[3] FindWeaponById
[4] UpdateWeapon
[5] FindAllWeapons

Soldiers actions:
[6] CreateSoldier
[7] DeleteSoldier
[8] FindSoldierById
[9] UpdateSoldier
[10] FindAllSoldiers

Team actions:
[11] CreateTeam
[12] DeleteTeam
[13] FindTeamById
[14] UpdateTeam
[15] RemoveCommanderFromTeam
[16] FindTeamByCommander
[17] FindAllTeams

Platoon actions:
[18] CreatePlatoon
[19] DeletePlatoon
[20] FindPlatoonById
[21] UpdatePlatoon
[22] RemoveCommanderFromPlatoon
[23] FindPlatoonByCommander
[24] FindAllPlatoons
```

Akcja: Find all weapons


```

Podaj akcję
FindWeaponById
ID broni
105
cze 14, 2020 1:09:49 PM org.hibernate.jpa.internal.util.LogHelper logPersistenceUnitInformation
INFO: HHH000204: Processing PersistenceUnitInfo [name: techobdb]
cze 14, 2020 1:09:49 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure
WARN: HHH10001002: Using Hibernate built-in connection pool (not for production use!)
cze 14, 2020 1:09:49 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001005: using driver [com.mysql.cj.jdbc.Driver] at URL [jdbc:mysql://localhost:3306/techobdb?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacy
cze 14, 2020 1:09:49 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001001: Connection properties: {user=ToAdmin, password=****}
cze 14, 2020 1:09:49 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001003: Autocommit mode: false
cze 14, 2020 1:09:49 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PooledConnections <init>
INFO: HHH000115: Hibernate connection pool size: 20 (min=1)
cze 14, 2020 1:09:49 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQL8Dialect
cze 14, 2020 1:09:49 PM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolatedConnection
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnectio
cze 14, 2020 1:09:49 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
Weapon(id=105, name=AKS-74, serialNumber=AS321, weaponType=ASSAULT_RIFLE)
Brak żołnierza
Lista dostępnych akcji:
Weapon actions:
    [1] CreateWeapon
    [2] DeleteWeapon
    [3] FindWeaponById
    [4] UpdateWeapon
    [5] FindAllWeapons

Soldiers actions:
    [6] CreateSoldier
    [7] DeleteSoldier
    [8] FindSoldierById
    [9] UpdateSoldier
    [10] FindAllSoldiers

Team actions:
    [11] CreateTeam
    [12] DeleteTeam
    [13] FindTeamById
    [14] UpdateTeam
    [15] RemoveCommanderFromTeam
    [16] FindTeamByCommander
    [17] FindAllTeams

Platoon actions:
    [18] CreatePlatoon

```

Akcja: Find weapon by ID

```

Problems @ Javadoc Declaration Console
Main [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (14 cze 2020, 13:03:55)

Podaj akcję
UpdateWeapon
ID broni
105
cze 14, 2020 1:10:34 PM org.hibernate.jpa.internal.util.LogHelper logPersistenceUnitInformation
INFO: HHH000204: Processing PersistenceUnitInfo [name: techobdb]
cze 14, 2020 1:10:34 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure
WARN: HHH10001002: Using Hibernate built-in connection pool (not for production use!)
cze 14, 2020 1:10:34 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001005: using driver [com.mysql.cj.jdbc.Driver] at URL [jdbc:mysql://localhost:3306/techobdb?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacy
cze 14, 2020 1:10:34 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001001: Connection properties: {user=ToAdmin, password=****}
cze 14, 2020 1:10:34 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001003: Autocommit mode: false
cze 14, 2020 1:10:34 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PooledConnections <init>
INFO: HHH000115: Hibernate connection pool size: 20 (min=1)
cze 14, 2020 1:10:34 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQL8Dialect
cze 14, 2020 1:10:34 PM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolatedConnection
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnectio
cze 14, 2020 1:10:34 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
Podaj nową nazwę.(Zostaw puste jeśli nie chcesz zmieniać)
PKM
Podaj nowy nr. seryjny.(Zostaw puste jeśli nie chcesz zmieniać)
Dostępne typy broni
0:ASSAULT_RIFLE
1:LMG
2:RPG
3:SNIPER_RIFLE
4:PISTOL
LMG
cze 14, 2020 1:10:53 PM org.hibernate.jpa.internal.util.LogHelper logPersistenceUnitInformation
INFO: HHH000204: Processing PersistenceUnitInfo [name: techobdb]
cze 14, 2020 1:10:53 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure
WARN: HHH10001002: Using Hibernate built-in connection pool (not for production use!)
cze 14, 2020 1:10:53 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001005: using driver [com.mysql.cj.jdbc.Driver] at URL [jdbc:mysql://localhost:3306/techobdb?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacy
cze 14, 2020 1:10:53 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001001: Connection properties: {user=ToAdmin, password=****}
cze 14, 2020 1:10:53 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001003: Autocommit mode: false
cze 14, 2020 1:10:53 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PooledConnections <init>
INFO: HHH000115: Hibernate connection pool size: 20 (min=1)
cze 14, 2020 1:10:53 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQL8Dialect
cze 14, 2020 1:10:53 PM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolatedConnection
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnectio
cze 14, 2020 1:10:53 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
Weapon(id=105, name=PKM, serialNumber=AS321, weaponType=LMG)
Brak żołnierza
Lista dostępnych akcji:
Weapon actions:
    [1] CreateWeapon
    [2] DeleteWeapon
    [3] FindWeaponById

```

Akcja: Update weapon wraz ze zmianami pokazanymi w find weapon by id.

```
Podaj akcję
AssignWeaponToSoldier
Podaj id żołnierza.(słowo <<cancel>> zavraca)
93
cze 14, 2020 1:14:38 PM org.hibernate.jpa.internal.util.LogHelper logPersistenceUnitInformation
INFO: HHH000204: Processing PersistenceUnitInfo [name: techobdb]
cze 14, 2020 1:14:38 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure
WARN: HHH10001002: Using Hibernate built-in connection pool (not for production use!)
cze 14, 2020 1:14:38 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001005: using driver [com.mysql.cj.jdbc.Driver] at URL [jdbc:mysql://localhost:3306/techobdb?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLe
cze 14, 2020 1:14:38 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001001: Connection properties: {user=ToAdmin, password=****}
cze 14, 2020 1:14:38 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001003: Autocommit mode: false
cze 14, 2020 1:14:38 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PooledConnections <init>
INFO: HHH000115: Hibernate connection pool size: 20 (min=1)
cze 14, 2020 1:14:38 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQL8Dialect
cze 14, 2020 1:14:38 PM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolatedConnection
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnecti
cze 14, 2020 1:14:38 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
ID broni
105
cze 14, 2020 1:14:48 PM org.hibernate.jpa.internal.util.LogHelper logPersistenceUnitInformation
INFO: HHH000204: Processing PersistenceUnitInfo [name: techobdb]
cze 14, 2020 1:14:48 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure
WARN: HHH10001002: Using Hibernate built-in connection pool (not for production use!)

cze 14, 2020 1:15:21 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
Weapon(id=105, name=PKM, serialNumber=AS321, weaponType=LMG)
Soldier(id=93, name=x, lastName=x, bloodType=A_POS, rank=CORPORAL)
Lista dostępnych akcji:
Weapon actions:
[1] CreateWeapon
```

Akcja: Assign weapon to soldier wraz z wynikiem pokazanym w show weapon by id

```
Podaj akcję
DeleteWeapon
ID broni
105
cze 14, 2020 1:25:27 PM org.hibernate.jpa.internal.util.LogHelper logPersistenceUnitInformation
INFO: HHH000204: Processing PersistenceUnitInfo [name: techobdb]
cze 14, 2020 1:25:27 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure
WARN: HHH10001002: Using Hibernate built-in connection pool (not for production use!)

INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
Weapon(id=64, name=k, serialNumber=k, weaponType=ASSAULT_RIFLE)
Weapon(id=86, name=ww, serialNumber=gggh, weaponType=RPG)
Weapon(id=103, name=PKM, serialNumber=1234, weaponType=LMG)
```

Lista dostępnych akcji:

Akcja: Delete weapon wraz z wynikiem działania w akcji find all weapons

```
Podaj akcję
FindPlatoonsWithoutCommander
cze 14, 2020 1:16:45 PM org.hibernate.jpa.internal.util.LogHelper logPersistenceUnitInformation
INFO: HHH000204: Processing PersistenceUnitInfo [name: techobdb]
cze 14, 2020 1:16:45 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure
WARN: HHH10001002: Using Hibernate built-in connection pool (not for production use!)
cze 14, 2020 1:16:45 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001005: using driver [com.mysql.cj.jdbc.Driver] at URL [jdbc:mysql://localhost:3306/techobdb?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLe
cze 14, 2020 1:16:45 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001001: Connection properties: {user=ToAdmin, password=****}
cze 14, 2020 1:16:45 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001003: Autocommit mode: false
cze 14, 2020 1:16:45 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PooledConnections <init>
INFO: HHH000115: Hibernate connection pool size: 20 (min=1)
cze 14, 2020 1:16:45 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQL8Dialect
cze 14, 2020 1:16:45 PM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolatedConnection
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnecti
cze 14, 2020 1:16:45 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
Platoon(super=Unit(id=90, number=1, commander=null))
Platoon(super=Unit(id=98, number=1, commander=null))
```

Akcja: Find platoons without commander

```
Podaj akcję
AssignCommanderToPlatoon
ID
90
cze 14, 2020 1:18:54 PM org.hibernate.jpa.internal.util.LogHelper logPersistenceUnitInformation
INFO: HHH000204: Processing PersistenceUnitInfo [name: techobdb]
cze 14, 2020 1:18:54 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure
WARN: HHH10001002: Using Hibernate built-in connection pool (not for production use!)
cze 14, 2020 1:18:54 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001005: using driver [com.mysql.cj.jdbc.Driver] at URL [jdbc:mysql://localhost:3306/techobdb?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC]
cze 14, 2020 1:18:54 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001001: Connection properties: {user=ToAdmin, password=****}
cze 14, 2020 1:18:54 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001003: Autocommit mode: false
cze 14, 2020 1:18:54 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PooledConnections <init>
INFO: HHH000115: Hibernate connection pool size: 20 (min=1)
cze 14, 2020 1:18:54 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQL8Dialect
cze 14, 2020 1:18:54 PM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolatedConnection
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnectionAccess]
cze 14, 2020 1:18:54 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
Podaj id dowódcy.(słowo <<cancel>> zwraca)
95
cze 14, 2020 1:19:03 PM org.hibernate.jpa.internal.util.LogHelper logPersistenceUnitInformation
INFO: HHH000204: Processing PersistenceUnitInfo [name: techobdb]
```

```
Podaj akcję
FindPlatoonById
ID
90
cze 14, 2020 1:19:49 PM org.hibernate.jpa.internal.util.LogHelper logPersistenceUnitInformation
INFO: HHH000204: Processing PersistenceUnitInfo [name: techobdb]
cze 14, 2020 1:19:49 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure
WARN: HHH10001002: Using Hibernate built-in connection pool (not for production use!)
cze 14, 2020 1:19:49 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001005: using driver [com.mysql.cj.jdbc.Driver] at URL [jdbc:mysql://localhost:3306/techobdb?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC]
cze 14, 2020 1:19:49 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001001: Connection properties: {user=ToAdmin, password=****}
cze 14, 2020 1:19:49 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001003: Autocommit mode: false
cze 14, 2020 1:19:49 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PooledConnections <init>
INFO: HHH000115: Hibernate connection pool size: 20 (min=1)
cze 14, 2020 1:19:49 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQL8Dialect
cze 14, 2020 1:19:49 PM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolatedConnection
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnectionAccess]
cze 14, 2020 1:19:49 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
Platoon(super=Unit(id=90, number=1, commander=Soldier(id=95, name=q, lastName=q, bloodType=A_NEG, rank=LT_1ST)))

-----Kompania:
Company(super=Unit(id=77, number=10, commander=Soldier(id=96, name=v, lastName=v, bloodType=A_POS, rank=CAPITAN)))

-----Drużyny:
Team(super=Unit(id=85, number=1, commander=Soldier(id=99, name=z, lastName=z, bloodType=A_NEG, rank=CORPORAL)))

Lista dostępnych akcji:
```

Akcja: Assign commander to platoon wraz z wynikiem działania w akcji find platoon by id

```
Podaj akcję
RemoveCommanderFromPlatoon
ID
90
cze 14, 2020 1:21:30 PM org.hibernate.jpa.internal.util.LogHelper logPersistenceUnitInformation
INFO: HHH000204: Processing PersistenceUnitInfo [name: techobdb]
cze 14, 2020 1:21:30 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
Platoon(super=Unit(id=90, number=1, commander=null))
Platoon(super=Unit(id=98, number=1, commander=null))
Lista dostępnych akcji:
```

Akcja: Remove commander from platoon wraz z wynikiem działania w akcji find platoons without commander

```

Podaj akcję
FindCompanyByCommander
Podaj imię
v
Podaj nazwisko
v
cze 14, 2020 1:24:14 PM org.hibernate.jpa.internal.util.LogHelper logPersistenceUnitInformation
INFO: HHH000204: Processing PersistenceUnitInfo [name: techobdb]
cze 14, 2020 1:24:14 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure
WARN: HHH10001002: Using Hibernate built-in connection pool (not for production use!)
cze 14, 2020 1:24:14 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001005: using driver [com.mysql.cj.jdbc.Driver] at URL [jdbc:mysql://localhost:3306/techobdb?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeMode=4&allowPublicKeyRetrieval=true]
cze 14, 2020 1:24:14 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001001: Connection properties: {user=ToAdmin, password=****}
cze 14, 2020 1:24:14 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001003: Autocommit mode: false
cze 14, 2020 1:24:14 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PooledConnections <init>
INFO: HHH000115: Hibernate connection pool size: 20 (min=1)
cze 14, 2020 1:24:14 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQL8Dialect
cze 14, 2020 1:24:14 PM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolatedConnection
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnectionAccess]
cze 14, 2020 1:24:14 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
cze 14, 2020 1:24:14 PM org.hibernate.jpa.internal.util.LogHelper logPersistenceUnitInformation
INFO: HHH000204: Processing PersistenceUnitInfo [name: techobdb]
cze 14, 2020 1:24:14 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure
WARN: HHH10001002: Using Hibernate built-in connection pool (not for production use!)
cze 14, 2020 1:24:14 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001005: using driver [com.mysql.cj.jdbc.Driver] at URL [jdbc:mysql://localhost:3306/techobdb?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeMode=4&allowPublicKeyRetrieval=true]
cze 14, 2020 1:24:14 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001001: Connection properties: {user=ToAdmin, password=****}
cze 14, 2020 1:24:14 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001003: Autocommit mode: false
cze 14, 2020 1:24:14 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PooledConnections <init>
INFO: HHH000115: Hibernate connection pool size: 20 (min=1)
cze 14, 2020 1:24:14 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQL8Dialect
cze 14, 2020 1:24:14 PM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolatedConnection
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnectionAccess]
cze 14, 2020 1:24:14 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
v v
Company(super=Unit(id=77, number=10, commander=Soldier(id=96, name=v, lastName=v, bloodType=A_POS, rank=CAPITAN)))
Lista dostępnych akcji:

```

Akcja: Find company by commander

7. Wnioski:

W ramach projektu stworzyliśmy aplikację do przeglądania i zarządzania bazą danych o tematyce militarnej.

Wykonany program pozwala posiada tekstowy interfejs użytkownika oraz bogatą funkcjonalność związaną z tematyką projektu.

Realizacja warstwy persystencji została wykonana przy wykorzystaniu frameworka Hibernate który pozwala na przełożenie klasy Encji na tabelę a instancje tej klasy na kolejne rekordy. Dzięki temu operując obiektami różnych klas możemy wprowadzać zmiany w bazie danych, skupiając się bardziej na logice biznesowej.

Aplikacja została stworzona wykorzystując wzorzec MVC oraz inne dobre praktyki w projektach informatycznych.

Cechuje się ona wysoką generycznością i reużywalnością komponentów a także wykorzystaniem zaawansowanych wzorców architektury.

Logika aplikacji skupiona jest w Akcjach. Są to klasy odpowiedzialne za pojedynczą operację jaką może wykonać użytkownik.

Akcje te wyświetlane są na liście do wyboru, a wywołanie którejs z nich powoduje uruchomienie logiki związanej z tą operacją.

Aby zapewnić Akcjom możliwość dostępu do danych, mają one referencje potrzebnych repozytoriów.

Repozytoria stanowią interfejs dostępu do bazy danych dla pojedynczej encji, dzięki któremu wykonuje się operacje CRUD.